(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 3 965 360 A1**

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
09.03.2022 Bulletin 2022/10

(21) Application number: **21186074.7**

(22) Date of filing: **16.07.2021**

(51) International Patent Classification (IPC):
H04L 9/32 *(2006.01)* ***H04L 9/08*** *(2006.01)*

(52) Cooperative Patent Classification (CPC):
H04L 9/3247; H04L 9/0825; H04L 9/0836; H04L 9/0877; H04L 9/3239

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
KH MA MD TN

(30) Priority: **08.09.2020 US 202017014600**

(71) Applicant: **INTEL Corporation**
Santa Clara, CA 95054 (US)

(72) Inventors:
- **MISOCZKI, RAFAEL**
  Hillsboro, OR Oregon 97124 (US)
- **REINDERS, Andrew H.**
  Portland, OR Oregon 97205 (US)
- **GHOSH, SANTOSH**
  Hillsboro, OR Oregon 97124 (US)
- **SASTRY, MANOJ**
  Portland, OR Oregon 97229 (US)

(74) Representative: **Goddar, Heinz J.**
Boehmert & Boehmert
Anwaltspartnerschaft mbB
Pettenkoferstrasse 22
80336 München (DE)

(54) STATE SYNCHRONIZATION FOR POST-QUANTUM SIGNING FACILITIES

(57) An apparatus comprises a plurality of hardware security modules, at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to generate a first plurality of pairs of cryptographic key pairs comprising a first plurality of private keys and a first plurality of public keys, forward the first plurality of public keys to a remote computing device, receive, from the remote computing device, a first plurality of ciphertexts, wherein each ciphertext in the plurality of ciphertexts represents an encryption of a cryptographic seed with a public key selected from the plurality of public keys, receive, from a subset of hardware security modules in the plurality of hardware security modules, a subset of private keys.

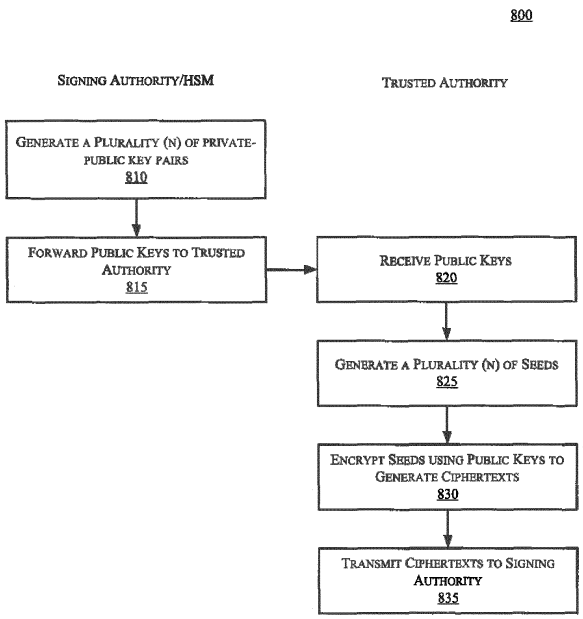

FIG. 8

EP 3 965 360 A1

Description

BACKGROUND

[0001] Subject matter described herein relates generally to the field of computer security and more particularly to code signing facilities for post-quantum cryptography secure hash-based signatures, including but not limited to the Extended Merkle Signature Scheme (XMSS) and Leighton/Micali Signature (LMS) hash-based signing and verification algorithms.

[0002] Existing public-key digital signature algorithms such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) are anticipated not to be secure against brute-force attacks based on algorithms such as Shor's algorithm using quantum computers. As a result, there are efforts underway in the cryptography research community and in various standards bodies to define new standards for algorithms that are secure against quantum computers.

[0003] Accordingly, techniques to manage the proper application of post-quantum signature schemes may find utility, e.g., in computer-based communication systems and methods.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The detailed description is described with reference to the accompanying figures.

Figs. 1A and 1B are schematic illustrations of a one-time hash-based signatures scheme and a multi-time hash-based signatures scheme, respectively.

Figs. 2A-2B are schematic illustrations of a one-time signature scheme and a multi-time signature scheme, respectively.

Fig. 3 is a schematic illustration of a signing device and a verifying device, in accordance with some examples.

Fig. 4A is a schematic illustration of a Merkle tree structure, in accordance with some examples.

Fig. 4B is a schematic illustration of a Merkle tree structure, in accordance with some examples.

Fig. 5 is a schematic illustration of a compute blocks in an architecture to implement a signature algorithm, in accordance with some examples.

Fig. 6A is a schematic illustration of a compute blocks in an architecture to implement signature generation in a signature algorithm, in accordance with some examples.

Fig. 6B is a schematic illustration of a compute blocks in an architecture to implement signature verification in a verification algorithm, in accordance with some examples.

Fig. 7 is a schematic illustration of compute blocks in an architecture to implement state synchronization for post-quantum signing facilities, in accordance with some examples.

Fig. 8 is a flowchart illustrating operations in a method to implement state synchronization for post-quantum signing facilities, in accordance with some examples.

Fig. 9 is a flowchart illustrating operations in a method to implement state synchronization for post-quantum signing facilities, in accordance with some examples.

Fig. 10 is a schematic illustration of a computing architecture which may be adapted to implement hardware acceleration in accordance with some examples.

DETAILED DESCRIPTION

[0005] Described herein are exemplary systems and methods to implement robust state synchronization for stateful hash-based signatures. In the following description, numerous specific details are set forth to provide a thorough understanding of various examples. However, it will be understood by those skilled in the art that the various examples may be practiced without the specific details. In other instances, well-known methods, procedures, components, and circuits have not been illustrated or described in detail so as not to obscure the examples.

[0006] As described briefly above, existing public-key digital signature algorithms such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) are anticipated not to be secure against brute-force attacks based on algorithms such as Shor's algorithm using quantum computers. Hash-based signatures, on the other hand, are expected to withstand attacks by quantum computers. One example of hash-based signature scheme is the eXtended Merkle Signature Scheme (XMSS). As used herein, the term XMSS shall refer to both the XMSS scheme and the XMSS-MT scheme.

[0007] An XMSS signature process implements a hash-based signature scheme using a one-time signature scheme such as a Winternitz one-time signature (WOTS) or a derivative there of (e.g., WOTS+) in combination with a secure hash algorithm (SHA) such as SHA2-256 as the primary underlying hash function. In some examples the XMSS signature/verification scheme may also use one or more of SHA2-512, SHA3-SHAKE-256 or SHA3-SHAKE-512 as secure

hash functions. XMSS-specific hash functions include a Pseudo-Random Function (PRF), a chain hash (F), a tree hash (H) and message hash function (H_{msg}). As used herein, the term WOTS shall refer to the WOTS signature scheme and or a derivative scheme such as WOTS+.

[0008] The Leighton/Micali signature (LMS) scheme is another hash-based signature scheme that uses Leighton/Micali one-time signatures (LM-OTS) as the one-time signature building block. LMS signatures are based on a SHA2-256 hash function.

[0009] An XMSS signature process comprises three major operations. The first major operation receives an input message (M) and a private key (sk) and utilizes a one-time signature algorithm (e.g., WOTS+) to generate a message representative (M') that encodes a public key (pk). In a 128-bit post quantum security implementation the input message M is subjected to a hash function and then divided into 67 message components (n bytes each), each of which are subjected to a hash chain function to generate the a corresponding 67 components of the digital signature. Each chain function invokes a series of underlying secure hash algorithms (SHA).

[0010] The second major operation is an L-Tree computation, which combines WOTS+ (or WOTS) public key components (n-bytes each) and produces a single n-byte value. For example, in the 128-bit post-quantum security there are 67 public key components, each of which invokes an underlying secure hash algorithm (SHA) that is performed on an input block.

[0011] The third major operation is a tree-hash operation, which constructs a Merkle tree. In an XMSS verification, an authentication path that is provided as part of the signature and the output of L-tree operation is processed by a tree-hash operation to generate the root node of the Merkle tree, which should correspond to the XMSS public key. For XMSS verification with 128-bit post-quantum security, traversing the Merkle tree comprises executing secure hash operations. In an XMSS verification, the output of the Tree-hash operation is compared with the known public key. If they match then the signature is accepted. By contrast, if they do not match then the signature is rejected.

[0012] An important limitation of all OTS algorithms, and many hash-based signature schemes built upon OTS techniques, is that use of any single private key more than once enables an attacker to forge signatures in the scheme. It is therefore imperative that systems which enable automated signing of code, as is common in Continuous Integration/Continuous Delivery (CI/CD) software development methodologies, single usage of an HBS signing key is guaranteed. Further, recovery from equipment failures or environmental conditions which may create a business continuity disruption, are handled in a way that maintains the guarantee of single-use for every private key. Design and construction of automated signing facilities must take into account both normal operations to ensure HBS signing keys are used only once, as well as exception or disaster conditions that could disrupt the normal flow or sequence of use of HBS private keys.

Post-Quantum Cryptography Overview

[0013] Post-Quantum Cryptography (also referred to as "quantum-proof', "quantum-safe", "quantum-resistant", or simply "PQC") takes a futuristic and realistic approach to cryptography. It prepares those responsible for cryptography as well as end-users to know the cryptography is outdated; rather, it needs to evolve to be able to successfully address the evolving computing devices into quantum computing and post-quantum computing.

[0014] It is well-understood that cryptography allows for protection of data that is communicated online between individuals and entities and stored using various networks. This communication of data can range from sending and receiving of emails, purchasing of goods or services online, accessing banking or other personal information using websites, etc.

[0015] Conventional cryptography and its typical factoring and calculating of difficult mathematical scenarios may not matter when dealing with quantum computing. These mathematical problems, such as discrete logarithm, integer factorization, and elliptic-curve discrete logarithm, etc., are not capable of withstanding an attack from a powerful quantum computer. Although any post-quantum cryptography could be built on the current cryptography, the novel approach would need to be intelligent, fast, and precise enough to resist and defeat any attacks by quantum computers

[0016] Figures 1A and 1B illustrate a one-time hash-based signatures scheme and a multi-time hash-based signatures scheme, respectively. As aforesaid, hash-based cryptography is based on cryptographic systems like Lamport signatures, Merkle Signatures, extended Merkle signature scheme (XMSS), and SPHINCs scheme, etc. With the advent of quantum computing and in anticipation of its growth, there have been concerns about various challenges that quantum computing could pose and what could be done to counter such challenges using the area of cryptography.

[0017] One area that is being explored to counter quantum computing challenges is hash-based signatures (HBS) since these schemes have been around for a long while and possess the necessarily basic ingredients to counter the quantum counting and post-quantum computing challenges. HBS schemes are regarded as fast signature algorithms working with fast platform secured-boot, which is regarded as the most resistant to quantum and post-quantum computing attacks.

[0018] For example, as illustrated with respect to Figure 1A, a scheme of HBS is shown that uses Merkle trees along with a one-time signature (OTS) scheme 100, such as using a private key to sign a message and a corresponding public

key to verify the OTS message, where a private key only signs a single message.

[0019] Similarly, as illustrated with respect to Figure 1B, another HBS scheme is shown, where this one relates to multi-time signatures (MTS) scheme 150, where a private key can sign multiple messages.

[0020] Figures 2A and 2B illustrate a one-time signature scheme and a multi-time signature scheme, respectively. Continuing with HBS-based OTS scheme 100 of Figure 1A and MTS scheme 150 of Figure 1B, Figure 2A illustrates Winternitz OTS scheme 200, which was offered by Robert Winternitz of Stanford Mathematics Department publishing as hw(x) as opposed to h(x)|h(y), while Figure 2B illustrates XMSS MTS scheme 250, respectively.

[0021] For example, WOTS scheme 200 of Fig. 2A provides for hashing and parsing of messages into M, with 67 integers between [0, 1, 2, ... , 15], such as private key, sk, 205, signature, s, 210, and public key, pk, 215, with each having 67 components of 32 bytes each.

[0022] Fig. 2B illustrates XMSS MTS scheme 250 that allows for a combination of WOTS scheme 200 of Figure 2A and XMSS scheme having XMSS Merkle tree. As discussed previously with respect to Figure 2A, WOTs scheme 200 is based on a one-time public key, pk, 215, having 67 components of 32 bytes each, that is then put through L-Tree compression algorithm 260 to offer WOTS compressed pk to take a place in the XMSS Merkle tree of XMSS scheme 255. It is contemplated that XMSS signature verification may include computing WOTS verification and checking to determine whether a reconstructed root node matches the XMSS public key, such as root node = XMSS public key.

Post-Quantum Cryptography

[0023] Fig. 3 is a schematic illustration of a high-level architecture of a secure environment 300 that includes a first device 310 and a second device 350, in accordance with some examples. Referring to Fig. 3, each of the first device 310 and the second device 350 may be embodied as any type of computing device capable of performing the functions described herein. For example, in some embodiments, each of the first device 310 and the second device 350 may be embodied as a laptop computer, tablet computer, notebook, netbook, Ultrabook™, a smartphone, cellular phone, wearable computing device, personal digital assistant, mobile Internet device, desktop computer, router, server, workstation, and/or any other computing/communication device.

[0024] First device 310 includes one or more processor(s) 320 and a memory 322 to store a private key 324. The processor(s) 320 may be embodied as any type of processor capable of performing the functions described herein. For example, the processor(s) 320 may be embodied as a single or multi-core processor(s), digital signal processor, micro-controller, or other processor or processing/controlling circuit. Similarly, the memory 322 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In operation, the memory 322 may store various data and software used during operation of the first device 310 such as operating systems, applications, programs, libraries, and drivers. The memory 322 is communicatively coupled to the processor(s) 320. In some examples the private key 324 may reside in a secure memory that may be part memory 322 or may be separate from memory 322.

[0025] First device 310 further comprises a signing facility 330 which comprises one or more hardware security module(s) 331 which includes memory 322, signature logic, and verification logic 336. Hash logic 332 is configured to hash (i.e., to apply a hash function to) a message (M) to generate a hash value (m') of the message M. Hash functions may include, but are not limited to, a secure hash function, e.g., secure hash algorithms SHA2-256 and/or SHA3-256, etc. SHA2-256 may comply and/or be compatible with Federal Information Processing Standards (FIPS) Publication 180-4, titled: "Secure Hash Standard (SHS)", published by National Institute of Standards and Technology (NIST) in March 2012, and/or later and/or related versions of this standard. SHA3-256 may comply and/or be compatible with FIPS Publication 202, titled: "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", published by NIST in August 2015, and/or later and/or related versions of this standard.

[0026] Signature logic 332 may be configured to generate a signature to be transmitted, i.e., a transmitted signature and/or to verify a signature. In instances in which the first device 310 is the signing device, the transmitted signature may include a number, L, of transmitted signature elements with each transmitted signature element corresponding to a respective message element. For example, for each message element, mi, signature logic 332 may be configured to perform a selected signature operation on each private key element, S_{ki} of the private key, S_k, a respective number of times related to a value of each message element, mi included in the message representative m'. For example, signature logic 332 may be configured to apply a selected hash function to a corresponding private key element, S_{ki}, mi times. In another example, signature logic 334 may be configured to apply a selected chain function (that contains a hash function) to a corresponding private key element, S_{ki}, mi times. The selected signature operations may, thus, correspond to a selected hash-based signature scheme.

[0027] Hash-based signature schemes may include, but are not limited to, a Winternitz (W) one time signature (OTS) scheme, an enhanced Winternitz OTS scheme (e.g., WOTS+), a Merkle many time signature scheme, an extended Merkle signature scheme (XMSS) and/or an extended Merkle multiple tree signature scheme (XMSS-MT), etc. Hash functions may include, but are not limited to SHA2-256 and/or SHA3-256, etc. For example, XMSS and/or XMSS-MT

may comply or be compatible with one or more Internet Engineering Task Force (IETF.RTM.) informational draft Internet notes, e.g., draft draft-irtf-cfrg-xmss-hash-based-signatures-00, titled "XMSS: Extended Hash-Based Signatures, released April 2015, by the Internet Research Task Force, Crypto Forum Research Group of the IETF.RTM. and/or later and/or related versions of this informational draft, such as draft draft-irtf-cfrg-xmss-hash-based-signatures-06, released June 2016.

[0028] Winternitz OTS is configured to generate a signature and to verify a received signature utilizing a hash function. Winternitz OTS is further configured to use the private key and, thus, each private key element, S_{ki}, one time. For example, Winternitz OTS may be configured to apply a hash function to each private key element, mi or N-m_i times to generate a signature and to apply the hash function to each received message element N-$m_{i'}$ or $m_{i'}$ times to generate a corresponding verification signature element. The Merkle many time signature scheme is a hash-based signature scheme that utilizes an OTS and may use a public key more than one time. For example, the Merkle signature scheme may utilize Winternitz OTS as the one-time signature scheme. WOTS+ is configured to utilize a family of hash functions and a chain function.

[0029] XMSS, WOTS+ and XMSS-MT are examples of hash-based signature schemes that utilize chain functions. Each chain function is configured to encapsulate a number of calls to a hash function and may further perform additional operations. The number of calls to the hash function included in the chain function may be fixed. Chain functions may improve security of an associated hash-based signature scheme. Hash-based signature balancing, as described herein, may similarly balance chain function operations.

[0030] Cryptography logic 340 is configured to perform various cryptographic and/or security functions on behalf of the signing device 310. In some embodiments, the cryptography logic 340 may be embodied as a cryptographic engine, an independent security co-processor of the signing device 310, a cryptographic accelerator incorporated into the processor(s) 320, or a standalone software/firmware. In some embodiments, the cryptography logic 340 may generate and/or utilize various cryptographic keys (e.g., symmetric/asymmetric cryptographic keys) to facilitate encryption, decryption, signing, and/or signature verification. Additionally, in some embodiments, the cryptography logic 340 may facilitate to establish a secure connection with remote devices over communication link. It should further be appreciated that, in some embodiments, the cryptography module 340 and/or another module of the first device 310 may establish a trusted execution environment or secure enclave within which a portion of the data described herein may be stored and/or a number of the functions described herein may be performed.

[0031] After the signature is generated as described above, the message, M, and signature may then be sent by first device 310, e.g., via communication logic 342, to second device 350 via network communication link 390. In an embodiment, the message, M, may not be encrypted prior to transmission. In another embodiment, the message, M, may be encrypted prior to transmission. For example, the message, M, may be encrypted by cryptography logic 340 to produce an encrypted message. The message may be received by communication logic 382 and decrypted by cryptographic logic 380.

[0032] Second device 350 may also include one or more processors 360 and a memory 362 to store a public key 364. As described above, the processor(s) 360 may be embodied as any type of processor capable of performing the functions described herein. For example, the processor(s) 360 may be embodied as a single or multi-core processor(s), digital signal processor, microcontroller, or other processor or processing/controlling circuit. Similarly, the memory 362 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In operation, the memory 362 may store various data and software used during operation of the second device 350 such as operating systems, applications, programs, libraries, and drivers. The memory 362 is communicatively coupled to the processor(s) 360.

[0033] In some examples the public key 364 may be provided to verifier device 350 in a previous exchange. The public key, p_k, is configured to contain a number L of public key elements, i.e., $p_k=[p_{k1}, \ldots, p_{kL}]$. The public key 364 may be stored, for example, to memory 362.

[0034] Second device 350 further comprises a signing facility 370 comprising one or more hardware security module 371 which includes hash logic 372, signature logic, and verification logic 376. As described above, hash logic 372 is configured to hash (i.e., to apply a hash function to) a message (M) to generate a hash message (m'). Hash functions may include, but are not limited to, a secure hash function, e.g., secure hash algorithms SHA2-256 and/or SHA3-256, etc. SHA2-256 may comply and/or be compatible with Federal Information Processing Standards (FIPS) Publication 180-4, titled: "Secure Hash Standard (SHS)", published by National Institute of Standards and Technology (NIST) in March 2012, and/or later and/or related versions of this standard. SHA3-256 may comply and/or be compatible with FIPS Publication 202, titled: "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", published by NIST in August 2015, and/or later and/or related versions of this standard.

[0035] In instances in which the second device is the verifying device, hardware security module 371 is configured to generate a verification signature based, at least in part, on the signature received from the first device and based, at least in part, on the received message representative (m'). For example, hardware security module 371 may configured to perform the same signature operations, i.e., apply the same hash function or chain function as applied by hash logic

332 of hardware security module 331, to each received message element a number, N-m_i, (or m_i'), times to yield a verification message element. Whether a verification signature, i.e., each of the L verification message elements, corresponds to a corresponding public key element, p_{ki}, may then be determined. For example, verification logic 376 may be configured to compare each verification message element to the corresponding public key element, p_{ki}. If each of the verification message element matches the corresponding public key element, p_{ki}, then the verification corresponds to success. In other words, if all of the verification message elements match the public key elements, p_{ki}, . . . , p_{kL}, then the verification corresponds to success. If any verification message element does not match the corresponding public key element, p_{ki}, then the verification corresponds to failure.

[0036] As described in greater detail below, in some examples the hardware security module 331 of the first device 310 includes one or more accelerators 338 that cooperate with the hash logic 332, signature logic 334 and/or verification logic 336 to accelerate authentication operations. Similarly, in some examples the hardware security module 371 of the second device 310 includes one or more accelerators 378 that cooperate with the hash logic 372, signature logic 374 and/or verification logic 376 to accelerate authentication operations. Examples of accelerators are described in the following paragraphs and with reference to the accompanying drawings.

[0037] The various modules of the environment 300 may be embodied as hardware, software, firmware, or a combination thereof. For example, the various modules, logic, and other components of the environment 300 may form a portion of, or otherwise be established by, the processor(s) 320 of first device 310 or processor(s) 360 of second device 350, or other hardware components of the devices As such, in some embodiments, one or more of the modules of the environment 300 may be embodied as circuitry or collection of electrical devices (e.g., an authentication circuitry, a cryptography circuitry, a communication circuitry, a signature circuitry, and/or a verification circuitry). Additionally, in some embodiments, one or more of the illustrative modules may form a portion of another module and/or one or more of the illustrative modules may be independent of one another.

[0038] Fig. 4A is a schematic illustration of a Merkle tree structure illustrating signing operations, in accordance with some examples. Referring to Fig. 4A, an XMSS signing operation requires the construction of a Merkle tree 400A using the local public key from each leaf WOTS node 410A to generate a global public key (PK) 420A. In some examples the authentication path and the root node value can be computed off-line such that these operations do not limit performance. Each WOTS node 410A has a unique secret key, "sk" which is used to sign a message only once. The XMSS signature consists of a signature generated for the input message and an authentication path of intermediate tree nodes to construct the root of the Merkle tree.

[0039] Fig. 4B is a schematic illustration of a Merkle tree structure 400B during verification, in accordance with some examples. During verification, the input message and signature are used to compute the local public key 420B of the WOTS node, which is further used to compute the tree root value using the authentication path. A successful verification will match the computed tree root value to the public key PK shared by the signing entity. The WOTS and L-Tree operations constitute on average 82% and 16% of XMSS sign/verify latency respectively, thus defining the overall performance of the authentication system. Described herein are various pre-computation techniques which may be implemented to speed-up WOTS and L-Tree operations, thereby improving XMSS performance. The techniques are applicable to the other hash options and scale well for both software and hardware implementations.

[0040] Fig. 5 is a schematic illustration of a compute blocks in an architecture 500 to implement a signature algorithm, in accordance with some examples. Referring to Fig. 5, the WOTS+ operation involves 67 parallel chains of 16 SHA2-256 HASH functions, each with the secret key sk[66:0] as input. Each HASH operation in the chain consists of 2 pseudo-random functions (PRF) using SHA2-256 to generate a bitmask and a key. The bitmask is XOR-ed with the previous hash and concatenated with the key as input message to a 3rd SHA2-256 hash operation. The 67×32-byte WOTS public key pk[66:0] is generated by hashing secret key sk across the 67 hash chains. Analogous functions are performed for SHAKE 128.

[0041] Fig. 6A is a schematic illustration of a compute blocks in an architecture 600A to implement signature generation in a signature algorithm, in accordance with some examples. As illustrated in Fig. 6A, for message signing, the input message is hashed and pre-processed to compute a 67×4-bit value, which is used as an index to choose an intermediate hash value in each chain.

[0042] Fig. 6B is a schematic illustration of a compute blocks in an architecture 600B to implement signature verification in a verification algorithm, in accordance with some examples. Referring to Fig. 6B, during verification, the message is again hashed to compute the signature indices and compute the remaining HASH operations in each chain to compute the WOTS public key pk. This value and the authentication path are used to compute the root of the Merkle tree and compare with the shared public key PK to verify the message.

Synchronization for Post-Quantum Signing Facilities

[0043] Hash-Based Signature (HBS) schemes such as the XMSS schemes described above are stateful, which means that some state (e.g., a counter) needs to be securely stored in between signature generations. If a signer reuses the

same counter (which means reusing the same one-time signing key), this exposes the system to forgeability attacks. Some digital signature signing facilities utilize multiple Hardware Security Modules (HSM) to improve availability of the signing system. If HBS schemes are integrated into these signing facilities, these signing facilities need to offer a mechanism to securely synchronize the state (e.g., a monotonic counter) across all HSM signers available in the facility.

[0044] Additionally, because HBS schemes in some renditions have a limited number of total signing operations based on the size of the tree, an attack on the signing system or an error in the signing system could cause a "wear-out" of the signing key where all possible counters are used up and the signing key cannot be used anymore. In systems where the life of a product is tied to the life of the HBS signing key, this can result in a denial of service of the product. Signing facilities must ensure such accidental and malicious wear-out of signing keys are prevented or their impacts are minimized.

[0045] Fig. 7 is a schematic illustration of compute blocks in an architecture to implement state synchronization for post-quantum signing authorities, in accordance with some examples. Referring to Fig. 7, in some examples signing facility 700 may comprise a computer readable memory block (or register file) 710 which may be used to store signature operation inputs and intermediate results for the signature operations, a state synchronization manager 720, a load balancer 722, a plurality of hardware security modules 730A, 730B,...730N, (collectively referred to herein by reference numeral 730) which are configured to compute signatures using a common XMSS key pair 740.

[0046] In some examples, signing facility 700 may be communicatively coupled to one or more trusted authorities (TA) 750A, 750B, ... 750N (collectively referred to herein by reference numeral 750). Trusted authorities 750 may comprise processing facilities capable to issue digital certificates that certifies the ownership of a public key by the owner named in the certificate.

[0047] In some examples, operations may be implemented to ensure that each hardware security module used in calculating digital signatures for a Merkle tree uses a unique state synchronization counter sequence to prevent different hardware security modules from using the same counter. Further, techniques described herein keep leaf nodes encrypted to prevent unauthorized usage and implement threshold encryption techniques (e.g., a majority vote rule) to decrypt nodes. In some examples each HSM 730 may generate a set of public-private key pairs. Encryption operations may require the entire set of public-private key pairs, while decryption may be performed using a subset of public-private key pairs, provided the subset includes a threshold number of key pairs.

[0048] Fig. 8 is a flowchart illustrating operations in a method 800 to implement state synchronization for post-quantum signing authorities, in accordance with some examples. In some examples the operations depicted in Fig. 8 may be implemented by processing circuitry in, or communicatively coupled to, one or more hardware security modules 730 in a signing facility 700 and one or more trusted authorities 750. In the example architecture depicted in Fig. 7, the operations of Fig. 8 may be implemented by processing circuitry in the state synchronization manger 720, alone or in combination with the hardware security modules 730.

[0049] Referring to Fig. 8, at operation 810, processing circuity in one or more hardware security modules 730 generates a plurality (n) of public-private key pairs. At operation 815 the public-private key pairs generated in operation 810 are forwarded to one or more trusted authorities 750.

[0050] At operation 820 the one or more trusted authorities 750 receives the public-private keys from the hardware security modules 730 and, at operation 825 the one or more trusted authorities 750 generates a corresponding plurality (n) of cryptographic seeds using the private-public key pairs received in operation 820. At operation 830 the one or more trusted authorities 750 encrypts the seeds generated in operation 825 using the public keys from the public-private key pairs received in operation 820 to generate a corresponding plurality (n) of ciphertexts of the form:

$$\text{EQ 1} \qquad ct_i = \text{Enc}(pk_{1i}, \ldots\ pk_{ni}, seed_i)$$

[0051] At operation 835 the plurality (n) of ciphertexts are transmitted back to the one or more hardware security modules 730 in the signing facility 700, which concludes setup operations.

[0052] Referring to Fig. 9, at operation 910, a number of hardware security modules 730 available to perform a digital signature process is determined. In some examples the state synchronization manager 720 may maintain a listing of the hardware security modules 730 in the signing facility that are operational and available to perform a digital signature process. At operation 915 one or more ciphertexts are received in the signing facility, as described with reference to operation 835.

[0053] At operation 920 one or more hardware security modules 730 is selected to perform the digital signature operations using the ciphertexts received in operation 915. In some examples, the state synchronization manager 720 may selected a subset comprising one or more hardware security modules 730 from the plurality of hardware security modules that are operational and available to execute digital signature operations. At operation 925 a subset of private keys is received. Further, in some examples the load balancer 722 may monitor the workload of the different hardware security modules 730 to facilitate selecting a subset of the hardware security modules such that the computing load is distributed between the available hardware security modules 730 in a fashion that smooths the work load assigned to

the hardware security modules 730.

[0054] At operation 935 the hardware security module 730 selected to implement the signature operations receives the subset of private keys of the other hardware security modules 730 in the signing facility 700. For example, if hardware security module 1 730A is selected to implement the signature operations then it will be provided the private keys of the other hardware security modules 730 in the signing facility 700. In some examples the private keys may be provided by the state synchronization manager 720. In other examples the private keys may be provided directly by the other hardware security modules 730.

[0055] At operation 930 it is determined whether the hardware security module 730 selected to implement the signature operations obtained a threshold number of private keys to be allowed to decrypt the seeds. In some examples a majority-vote rule is applied, such that the hardware security module 730 selected to implement the signature operations must have obtained the private keys from a majority of the hardware security modules 730 in the signing facility, such that the threshold may be variable as a function of the number of hardware security modules 730 operating in the signing facility 700. In other examples, the threshold may be a static threshold that represents a fixed number of hardware security modules 730.

[0056] If, at operation 930, the hardware security module 730 selected to implement the signature operations obtains private keys from a number of hardware security modules 730 that exceeds the threshold, then control passes to operation 935 and the hardware security module 730 generates a first signal and, in response to the first signal, at operation 940 the hardware security module 730 decrypts the cryptographic seed and at operation 945 the hardware security module 730 generates the leaf nodes for the Merkle tree.

[0057] By contrast, if at operation 930 the hardware security module 730 selected to implement the signature operations fails to obtain private keys from a number of hardware security modules 730 that exceeds the threshold, then control passes to operation 950 and the hardware security module 730 generates a second signal and, in response to the second signal, at operation 955 the hardware security module 730 generates an error message indicating that the hardware security module 730 cannot generate the leaf nodes for the Merkle tree.

[0058] Thus, the operations described herein ensure that a rogue or malfunctioning hardware security module is not able to use unauthorized leaf nodes of the Merkle tree since decryption requires explicit authorization given by at least (k-1) other hardware security modules, where k represents the number of hardware security modules 730 in the signing facility 700. Further, techniques described herein require a central certificate authority only during the setup operations depicted in Fig. 8. It will be understood that a variety of different rules may be used to determine if a hardware security module 730 is behaving maliciously or defectively.

[0059] Fig. 10 illustrates an embodiment of an exemplary computing architecture that may be suitable for implementing various embodiments as previously described. In various embodiments, the computing architecture 1000 may comprise or be implemented as part of an electronic device. In some embodiments, the computing architecture 1000 may be representative, for example of a computer system that implements one or more components of the operating environments described above. In some embodiments, computing architecture 1000 may be representative of one or more portions or components of a digital signature signing system that implement one or more techniques described herein. The embodiments are not limited in this context.

[0060] As used in this application, the terms "system" and "component" and "module" are intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution, examples of which are provided by the exemplary computing architecture 1000. For example, a component can be, but is not limited to being, a process running on a processor, a processor, a hard disk drive, multiple storage drives (of optical and/or magnetic storage medium), an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a server and the server can be a component. One or more components can reside within a process and/or thread of execution, and a component can be localized on one computer and/or distributed between two or more computers. Further, components may be communicatively coupled to each other by various types of communications media to coordinate operations. The coordination may involve the uni-directional or bi-directional exchange of information. For instance, the components may communicate information in the form of signals communicated over the communications media. The information can be implemented as signals allocated to various signal lines. In such allocations, each message is a signal. Further embodiments, however, may alternatively employ data messages. Such data messages may be sent across various connections. Exemplary connections include parallel interfaces, serial interfaces, and bus interfaces.

[0061] The computing architecture 1000 includes various common computing elements, such as one or more processors, multi-core processors, co-processors, memory units, chipsets, controllers, peripherals, interfaces, oscillators, timing devices, video cards, audio cards, multimedia input/output (I/O) components, power supplies, and so forth. The embodiments, however, are not limited to implementation by the computing architecture 1000.

[0062] As shown in Figure 10, the computing architecture 1000 includes one or more processors 1002 and one or more graphics processors 1008, and may be a single processor desktop system, a multiprocessor workstation system, or a server system having a large number of processors 1002 or processor cores 1007. In on embodiment, the system

1000 is a processing platform incorporated within a system-on-a-chip (SoC or SOC) integrated circuit for use in mobile, handheld, or embedded devices.

[0063] An embodiment of system 1000 can include, or be incorporated within a server-based gaming platform, a game console, including a game and media console, a mobile gaming console, a handheld game console, or an online game console. In some embodiments system 1000 is a mobile phone, smart phone, tablet computing device or mobile Internet device. Data processing system 1000 can also include, couple with, or be integrated within a wearable device, such as a smart watch wearable device, smart eyewear device, augmented reality device, or virtual reality device. In some embodiments, data processing system 1000 is a television or set top box device having one or more processors 1002 and a graphical interface generated by one or more graphics processors 1008.

[0064] In some embodiments, the one or more processors 1002 each include one or more processor cores 1007 to process instructions which, when executed, perform operations for system and user software. In some embodiments, each of the one or more processor cores 1007 is configured to process a specific instruction set 1009. In some embodiments, instruction set 1009 may facilitate Complex Instruction Set Computing (CISC), Reduced Instruction Set Computing (RISC), or computing via a Very Long Instruction Word (VLIW). Multiple processor cores 1007 may each process a different instruction set 1009, which may include instructions to facilitate the emulation of other instruction sets. Processor core 1007 may also include other processing devices, such a Digital Signal Processor (DSP).

[0065] In some embodiments, the processor 1002 includes cache memory 1004. Depending on the architecture, the processor 1002 can have a single internal cache or multiple levels of internal cache. In some embodiments, the cache memory is shared among various components of the processor 1002. In some embodiments, the processor 1002 also uses an external cache (e.g., a Level-3 (L3) cache or Last Level Cache (LLC)) (not shown), which may be shared among processor cores 1007 using known cache coherency techniques. A register file 1006 is additionally included in processor 1002 which may include different types of registers for storing different types of data (e.g., integer registers, floating point registers, status registers, and an instruction pointer register). Some registers may be general-purpose registers, while other registers may be specific to the design of the processor 1002.

[0066] In some embodiments, one or more processor(s) 1002 are coupled with one or more interface bus(es) 1010 to transmit communication signals such as address, data, or control signals between processor 1002 and other components in the system. The interface bus 1010, in one embodiment, can be a processor bus, such as a version of the Direct Media Interface (DMI) bus. However, processor busses are not limited to the DMI bus, and may include one or more Peripheral Component Interconnect buses (e.g., PCI, PCI Express), memory busses, or other types of interface busses. In one embodiment the processor(s) 1002 include an integrated memory controller 1016 and a platform controller hub 1030. The memory controller 1016 facilitates communication between a memory device and other components of the system 1000, while the platform controller hub (PCH) 1030 provides connections to I/O devices via a local I/O bus.

[0067] Memory device 1020 can be a dynamic random-access memory (DRAM) device, a static random-access memory (SRAM) device, flash memory device, phase-change memory device, or some other memory device having suitable performance to serve as process memory. In one embodiment the memory device 1020 can operate as system memory for the system 1000, to store data 1022 and instructions 1021 for use when the one or more processors 1002 executes an application or process. Memory controller hub 1016 also couples with an optional external graphics processor 1012, which may communicate with the one or more graphics processors 1008 in processors 1002 to perform graphics and media operations. In some embodiments a display device 1011 can connect to the processor(s) 1002. The display device 1011 can be one or more of an internal display device, as in a mobile electronic device or a laptop device or an external display device attached via a display interface (e.g., DisplayPort, etc.). In one embodiment the display device 1011 can be a head mounted display (HMD) such as a stereoscopic display device for use in virtual reality (VR) applications or augmented reality (AR) applications.

[0068] In some embodiments the platform controller hub 1030 enables peripherals to connect to memory device 1020 and processor 1002 via a high-speed I/O bus. The I/O peripherals include, but are not limited to, an audio controller 1046, a network controller 1034, a firmware interface 1028, a wireless transceiver 1026, touch sensors 1025, a data storage device 1024 (e.g., hard disk drive, flash memory, etc.). The data storage device 1024 can connect via a storage interface (e.g., SATA) or via a peripheral bus, such as a Peripheral Component Interconnect bus (e.g., PCI, PCI Express). The touch sensors 1025 can include touch screen sensors, pressure sensors, or fingerprint sensors. The wireless transceiver 1026 can be a Wi-Fi transceiver, a Bluetooth transceiver, or a mobile network transceiver such as a 3G, 4G, or Long Term Evolution (LTE) transceiver. The firmware interface 1028 enables communication with system firmware, and can be, for example, a unified extensible firmware interface (UEFI). The network controller 1034 can enable a network connection to a wired network. In some embodiments, a high-performance network controller (not shown) couples with the interface bus 1010. The audio controller 1046, in one embodiment, is a multi-channel high definition audio controller. In one embodiment the system 1000 includes an optional legacy I/O controller 1040 for coupling legacy (e.g., Personal System 2 (PS/2)) devices to the system. The platform controller hub 1030 can also connect to one or more Universal Serial Bus (USB) controllers 1042 connect input devices, such as keyboard and mouse 1043 combinations, a camera 1244, or other USB input devices.

[0069] The following pertains to further examples.

Example 1 is an apparatus, comprising a computer readable memory; a plurality of hardware security modules, at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to generate a first plurality of pairs of cryptographic key pairs comprising a first plurality of private keys and a first plurality of public keys; forward the first plurality of public keys to a remote computing device; receive, from the remote computing device, a first plurality of ciphertexts, wherein each ciphertext in the plurality of ciphertexts represents an encryption of a cryptographic seed with a public key selected from the plurality of public keys; receive, from a subset of hardware security modules in the plurality of hardware security modules, a subset of private keys; and generate at least one of a first signal when the subset of private keys comprises a number of private keys that exceeds a threshold; or a second signal when the subset of private keys comprises a number of private keys that does not exceed a threshold.

In Example 2, the subject matter of Example 1 can optionally include an arrangement wherein the at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to generate an error message when the subset of private keys comprises a number of private keys that does not exceed the threshold.

In Example 3, the subject matter of any one of Examples 1-2 can optionally include an arrangement wherein the at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to decrypt the cryptographic seed when the subset of private keys comprises a number of private keys that exceeds the threshold.

In Example 4, the subject matter of any one of Examples 1-3 can optionally include an arrangement wherein the at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to generate a first plurality of digital signatures for a first plurality of leaf nodes in a Merkle tree.

In Example 5, the subject matter of any one of Examples 1-4 can optionally include an arrangement wherein the threshold is a static threshold that represents a fixed number of hardware security modules.

In Example 6, the subject matter of any one of Examples 1-5 can optionally an arrangement wherein the threshold is a dynamic threshold that represents a variable number of hardware security modules.

In Example 7, the subject matter of any one of Examples 1-6 can optionally include further comprising a state synchronization manager comprising a load balancer to select one of the first hardware security module or a second hardware security module to generate a signature.

Example 8 is a computer-based method, comprising selecting, from a plurality of hardware security modules in a signing facility, a set of hardware security modules to be assigned to a digital signature process, the set of hardware security modules comprising at least a first hardware security module; and in the at least a first hardware security module generating a first plurality of pairs of cryptographic key pairs comprising a first plurality of private keys and a first plurality of public keys; forwarding the first plurality of public keys to a remote computing device; receiving, from the remote computing device, a first plurality of ciphertexts, wherein each ciphertext in the plurality of ciphertexts represents an encryption of a cryptographic seed with a public key selected from the plurality of public keys; receiving, from a subset of hardware security modules in the plurality of hardware security modules, a subset of private keys; and generating at least one of

a first signal when the subset of private keys comprises a number of private keys that exceeds a threshold; or a second signal when the subset of private keys comprises a number of private keys that exceeds a threshold.

In Example 9, the subject matter of Example 8 can optionally include generating an error message when the subset of private keys comprises a number of private keys that does not exceed the threshold.

In Example 10, the subject matter of any one of Examples 9 can optionally include decrypting the cryptographic seed when the subset of private keys comprises a number of private keys that exceeds the threshold.

In Example 11, the subject matter of any one of Examples 9-10 can optionally include generating a first plurality of digital signatures for a first plurality of leaf nodes in a Merkle tree.

In Example 12, the subject matter of any one of Examples 9-11 can optionally include an arrangement wherein the threshold is a static threshold that represents a fixed number of hardware security modules.

In Example 13, the subject matter of any one of Examples 9-12 can optionally include an arrangement wherein the threshold is a dynamic threshold that represents a variable number of hardware security modules.

In Example 14, the subject matter of any one of Examples 9-13 can optionally include selecting one of the first hardware security module or a second hardware security module to generate a signature.

Example 15 is a non-transitory computer readable medium comprising instructions which, when executed by a processor, configure the processor to select, from a plurality of hardware security modules in a signing facility, a set of hardware security modules to be assigned to a digital signature process, the set of hardware security modules comprising at least a first hardware security module; and in the at least a first hardware security module generate a first plurality of pairs of cryptographic key pairs comprising a first plurality of private keys and a first plurality of

public keys; forward the first plurality of public keys to a remote computing device; receive, from the remote computing device, a first plurality of ciphertexts, wherein each ciphertext in the plurality of ciphertexts represents an encryption of a cryptographic seed with a public key selected from the plurality of public keys; receive, from a subset of hardware security modules in the plurality of hardware security modules, a subset of private keys; and generating at least one of a first signal when the subset of private keys comprises a number of private keys that exceeds a threshold; or a second signal when the subset of private keys comprises a number of private keys that exceeds a threshold.

In Example 16, the subject matter of Example 15 can optionally include the subject matter of claim 15, comprising instruction to generate an error message when the subset of private keys comprises a number of private keys that does not exceed the threshold.

In Example 17, the subject matter of any one of Examples 15-16 can optionally include instructions to decrypt the cryptographic seed when the subset of private keys comprises a number of private keys that exceeds the threshold.

In Example 18, the subject matter of any one of Examples 15-17 can optionally include instructions to generate a first plurality of digital signatures for a first plurality of leaf nodes in a Merkle tree.

In Example 19, the subject matter of any one of Examples 15-18 can optionally include an arrangement wherein the threshold is a static threshold that represents a fixed number of hardware security modules.

In Example 20, the subject matter of any one of Examples 15-19 can optionally include instructions to.

In Example 21, the subject matter of any one of Examples 15-20 can optionally include instructions to select one of the first hardware security module or a second hardware security module to generate a signature.

[0070] The above Detailed Description includes references to the accompanying drawings, which form a part of the Detailed Description. The drawings show, by way of illustration, specific embodiments that may be practiced. These embodiments are also referred to herein as "examples." Such examples may include elements in addition to those shown or described. However, also contemplated are examples that include the elements shown or described. Moreover, also contemplated are examples using any combination or permutation of those elements shown or described (or one or more aspects thereof), either with respect to a particular example (or one or more aspects thereof), or with respect to other examples (or one or more aspects thereof) shown or described herein.

[0071] Publications, patents, and patent documents referred to in this document are incorporated by reference herein in their entirety, as though individually incorporated by reference. In the event of inconsistent usages between this document and those documents so incorporated by reference, the usage in the incorporated reference(s) are supplementary to that of this document; for irreconcilable inconsistencies, the usage in this document controls.

[0072] In this document, the terms "a" or "an" are used, as is common in patent documents, to include one or more than one, independent of any other instances or usages of "at least one" or "one or more." In addition "a set of" includes one or more elements. In this document, the term "or" is used to refer to a nonexclusive or, such that "A or B" includes "A but not B," "B but not A," and "A and B," unless otherwise indicated. In the appended claims, the terms "including" and "in which" are used as the plain-English equivalents of the respective terms "comprising" and "wherein." Also, in the following claims, the terms "including" and "comprising" are open-ended; that is, a system, device, article, or process that includes elements in addition to those listed after such a term in a claim are still deemed to fall within the scope of that claim. Moreover, in the following claims, the terms "first," "second," "third," etc. are used merely as labels, and are not intended to suggest a numerical order for their objects.

[0073] The terms "logic instructions" as referred to herein relates to expressions which may be understood by one or more machines for performing one or more logical operations. For example, logic instructions may comprise instructions which are interpretable by a processor compiler for executing one or more operations on one or more data objects. However, this is merely an example of machine-readable instructions and examples are not limited in this respect.

[0074] The terms "computer readable medium" as referred to herein relates to media capable of maintaining expressions which are perceivable by one or more machines. For example, a computer readable medium may comprise one or more storage devices for storing computer readable instructions or data. Such storage devices may comprise storage media such as, for example, optical, magnetic or semiconductor storage media. However, this is merely an example of a computer readable medium and examples are not limited in this respect.

[0075] The term "logic" as referred to herein relates to structure for performing one or more logical operations. For example, logic may comprise circuitry which provides one or more output signals based upon one or more input signals. Such circuitry may comprise a finite state machine which receives a digital input and provides a digital output, or circuitry which provides one or more analog output signals in response to one or more analog input signals. Such circuitry may be provided in an application specific integrated circuit (ASIC) or field programmable gate array (FPGA). Also, logic may comprise machine-readable instructions stored in a memory in combination with processing circuitry to execute such machine-readable instructions. However, these are merely examples of structures which may provide logic and examples are not limited in this respect.

[0076] Some of the methods described herein may be embodied as logic instructions on a computer-readable medium. When executed on a processor, the logic instructions cause a processor to be programmed as a special-purpose machine

that implements the described methods. The processor, when configured by the logic instructions to execute the methods described herein, constitutes structure for performing the described methods. Alternatively, the methods described herein may be reduced to logic on, e.g., a field programmable gate array (FPGA), an application specific integrated circuit (ASIC) or the like.

[0077] In the description and claims, the terms coupled and connected, along with their derivatives, may be used. In particular examples, connected may be used to indicate that two or more elements are in direct physical or electrical contact with each other. Coupled may mean that two or more elements are in direct physical or electrical contact. However, coupled may also mean that two or more elements may not be in direct contact with each other, but yet may still cooperate or interact with each other.

[0078] Reference in the specification to "one example" or "some examples" means that a particular feature, structure, or characteristic described in connection with the example is included in at least an implementation. The appearances of the phrase "in one example" in various places in the specification may or may not be all referring to the same example.

[0079] The above description is intended to be illustrative, and not restrictive. For example, the above-described examples (or one or more aspects thereof) may be used in combination with others. Other embodiments may be used, such as by one of ordinary skill in the art upon reviewing the above description. The Abstract is to allow the reader to quickly ascertain the nature of the technical disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims. Also, in the above Detailed Description, various features may be grouped together to streamline the disclosure. However, the claims may not set forth every feature disclosed herein as embodiments may feature a subset of said features. Further, embodiments may include fewer features than those disclosed in a particular example. Thus, the following claims are hereby incorporated into the Detailed Description, with each claim standing on its own as a separate embodiment. The scope of the embodiments disclosed herein is to be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

[0080] Although examples have been described in language specific to structural features and/or methodological acts, it is to be understood that claimed subject matter may not be limited to the specific features or acts described. Rather, the specific features and acts are disclosed as sample forms of implementing the claimed subject matter.

Claims

1. An apparatus, comprising:

a computer readable memory;
a plurality of hardware security modules, at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to:

generate a first plurality of pairs of cryptographic key pairs comprising a first plurality of private keys and a first plurality of public keys;
forward the first plurality of public keys to a remote computing device;
receive, from the remote computing device, a first plurality of ciphertexts, wherein each ciphertext in the plurality of ciphertexts represents an encryption of a cryptographic seed with a public key selected from the plurality of public keys;
receive, from a subset of hardware security modules in the plurality of hardware security modules, a subset of private keys; and
generate at least one of:

a first signal when the subset of private keys comprises a number of private keys that exceeds a threshold; or
a second signal when the subset of private keys comprises a number of private keys that does not exceed a threshold.

2. The apparatus of claim 1, wherein the at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to:
generate an error message when the subset of private keys comprises a number of private keys that does not exceed the threshold.

3. The apparatus of any one of claims 1-2, wherein the at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to:

decrypt the cryptographic seed when the subset of private keys comprises a number of private keys that exceeds the threshold.

4. The apparatus of any one of claims 1-3, wherein the at least a first hardware security module in the plurality of hardware security modules comprising processing circuitry to:
generate a first plurality of digital signatures for a first plurality of leaf nodes in a Merkle tree.

5. The apparatus of any one of claims 1-4, wherein the threshold is a static threshold that represents a fixed number of hardware security modules.

6. The apparatus of any one of claims 1-5, wherein the threshold is a dynamic threshold that represents a variable number of hardware security modules.

7. The apparatus of any one of claims 1 -6, further comprising a state synchronization manager comprising:
a load balancer to select one of the first hardware security module or a second hardware security module to generate a signature.

8. A computer-based method, comprising:

selecting, from a plurality of hardware security modules in a signing facility, a set of hardware security modules to be assigned to a digital signature process, the set of hardware security modules comprising at least a first hardware security module; and
in the at least a first hardware security module:

generating a first plurality of pairs of cryptographic key pairs comprising a first plurality of private keys and a first plurality of public keys;
forwarding the first plurality of public keys to a remote computing device;
receiving, from the remote computing device, a first plurality of ciphertexts, wherein each ciphertext in the plurality of ciphertexts represents an encryption of a cryptographic seed with a public key selected from the plurality of public keys;
receiving, from a subset of hardware security modules in the plurality of hardware security modules, a subset of private keys; and
generating at least one of:

a first signal when the subset of private keys comprises a number of private keys that exceeds a threshold; or
a second signal when the subset of private keys comprises a number of private keys that exceeds a threshold.

9. The method of claim 8, further comprising:
generating an error message when the subset of private keys comprises a number of private keys that does not exceed the threshold.

10. The method of any one of claims 8-9, further comprising:
decrypting the cryptographic seed when the subset of private keys comprises a number of private keys that exceeds the threshold.

11. The method of any one of claims 8-10, further comprising:
generating a first plurality of digital signatures for a first plurality of leaf nodes in a Merkle tree.

12. The method of any one of claims 8-11, wherein the threshold is a static threshold that represents a fixed number of hardware security modules.

13. The method of any one of claims 8-12, wherein the threshold is a dynamic threshold that represents a variable number of hardware security modules.

14. The method of any one of claims 8-13, further comprising selecting one of the first hardware security module or a second hardware security module to generate a signature.

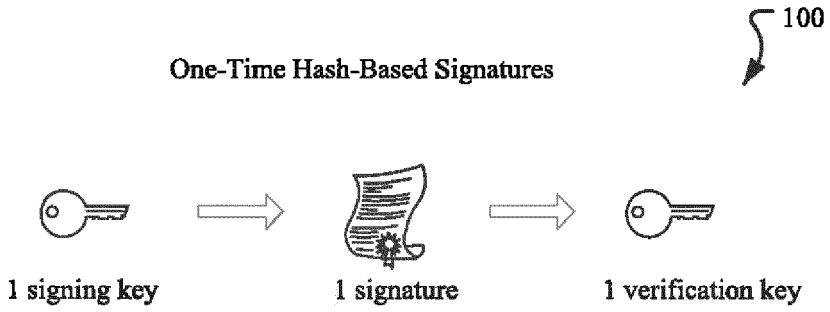
100
One-Time Hash-Based Signatures
1 signing key
1 signature
1 verification key
A private key must only sign a single message

FIG. 1A

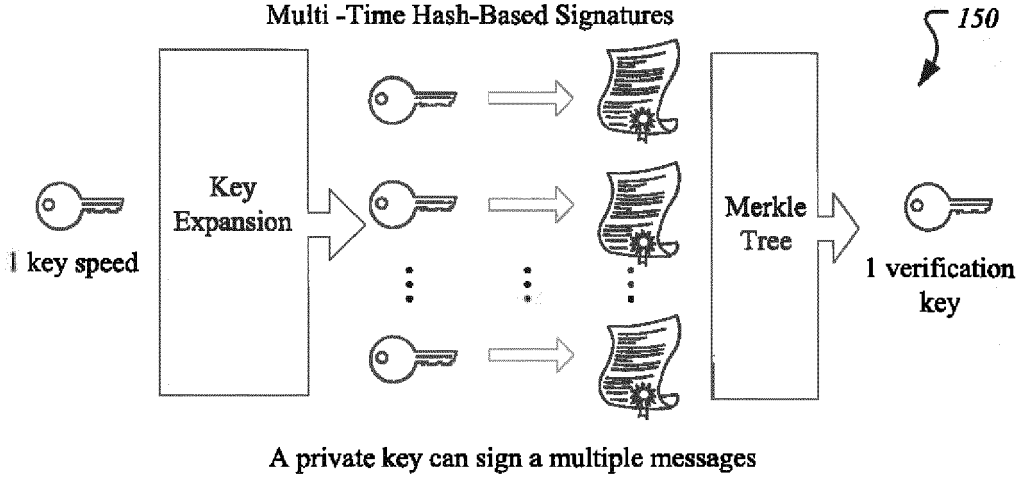
Multi -Time Hash-Based Signatures
150
1 key speed
Key Expansion
Merkle Tree
1 verification key
A private key can sign a multiple messages

FIG. 1B

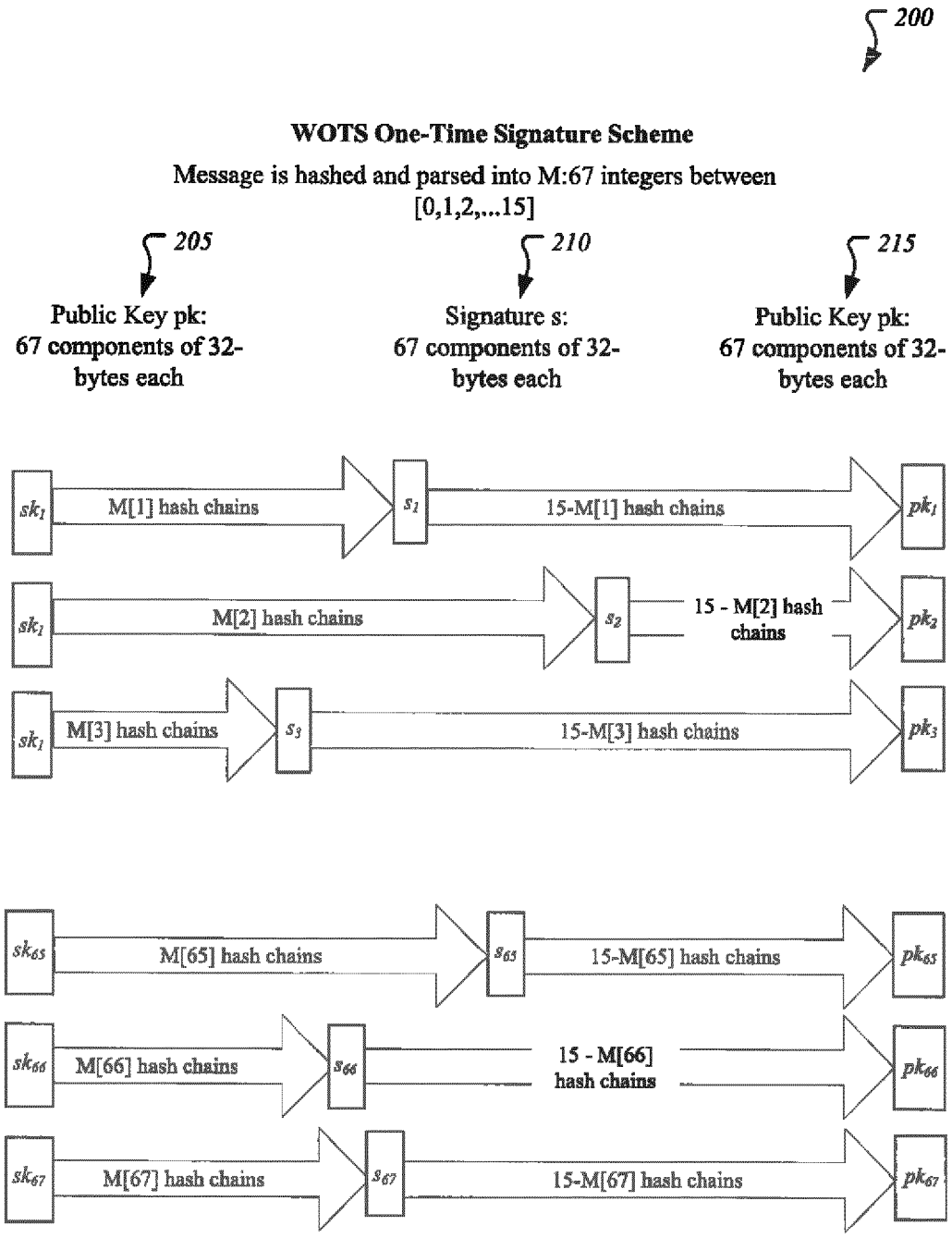
200
WOTS One-Time Signature Scheme
Message is hashed and parsed into M:67 integers between [0,1,2,...15]
205
210
215
Public Key pk: 67 components of 32-bytes each
Signature s: 67 components of 32-bytes each
Public Key pk: 67 components of 32-bytes each
sk1
M[1] hash chains
s1
15-M[1] hash chains
pk1
sk1
M[2] hash chains
s2
15 - M[2] hash chains
pk2
sk1
M[3] hash chains
s3
15-M[3] hash chains
pk3
sk65
M[65] hash chains
s65
15-M[65] hash chains
pk65
sk66
M[66] hash chains
s66
15 - M[66] hash chains
pk66
sk67
M[67] hash chains
s67
15-M[67] hash chains
pk67

FIG. 2A

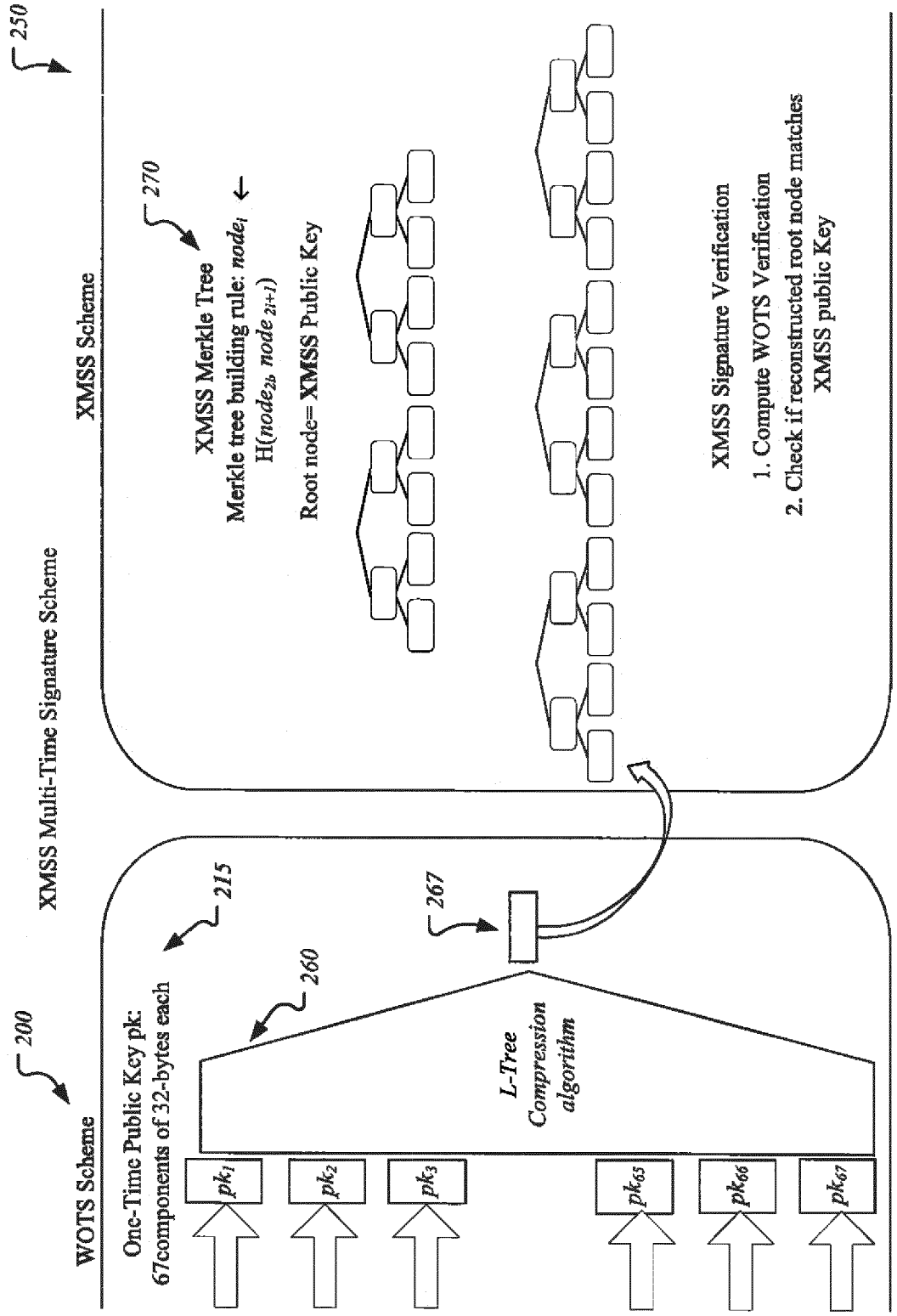
250
XMSS Multi-Time Signature Scheme
XMSS Scheme
270
XMSS Merkle Tree
Merkle tree building rule: $node_i$ ← H($node_{2i}$, $node_{2i+1}$)
Root node= XMSS Public Key
XMSS Signature Verification
1. Compute WOTS Verification
2. Check if reconstructed root node matches XMSS public Key
200
WOTS Scheme
One-Time Public Key pk: 67components of 32-bytes each
215
260
267
L-Tree Compression algorithm
pk_1
pk_2
pk_3
pk_{65}
pk_{66}
pk_{67}

FIG. 2B

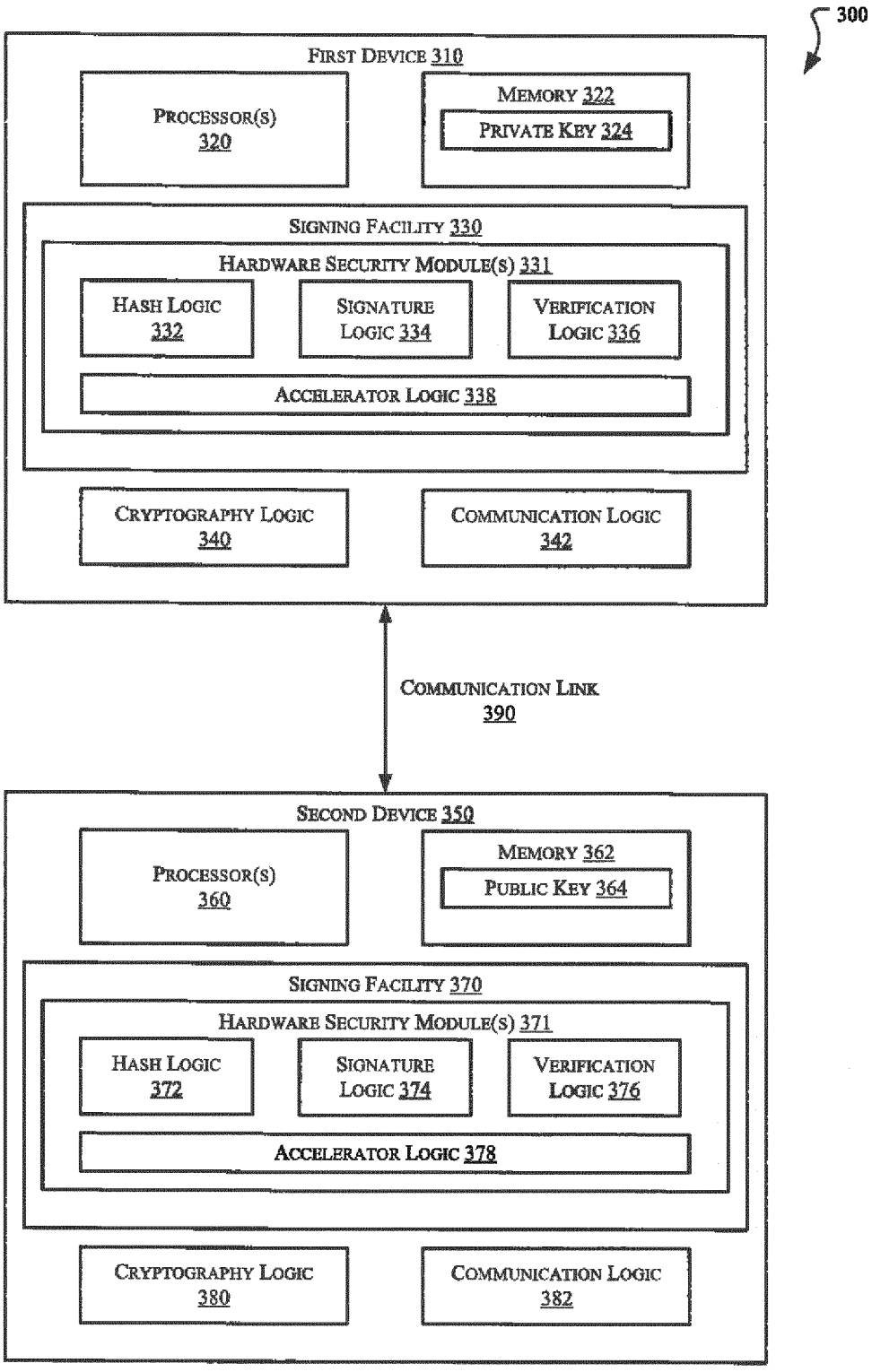
300
First Device 310
Processor(s) 320
Memory 322
Private Key 324
Signing Facility 330
Hardware Security Module(s) 331
Hash Logic 332
Signature Logic 334
Verification Logic 336
Accelerator Logic 338
Cryptography Logic 340
Communication Logic 342
Communication Link 390
Second Device 350
Processor(s) 360
Memory 362
Public Key 364
Signing Facility 370
Hardware Security Module(s) 371
Hash Logic 372
Signature Logic 374
Verification Logic 376
Accelerator Logic 378
Cryptography Logic 380
Communication Logic 382

FIG. 3

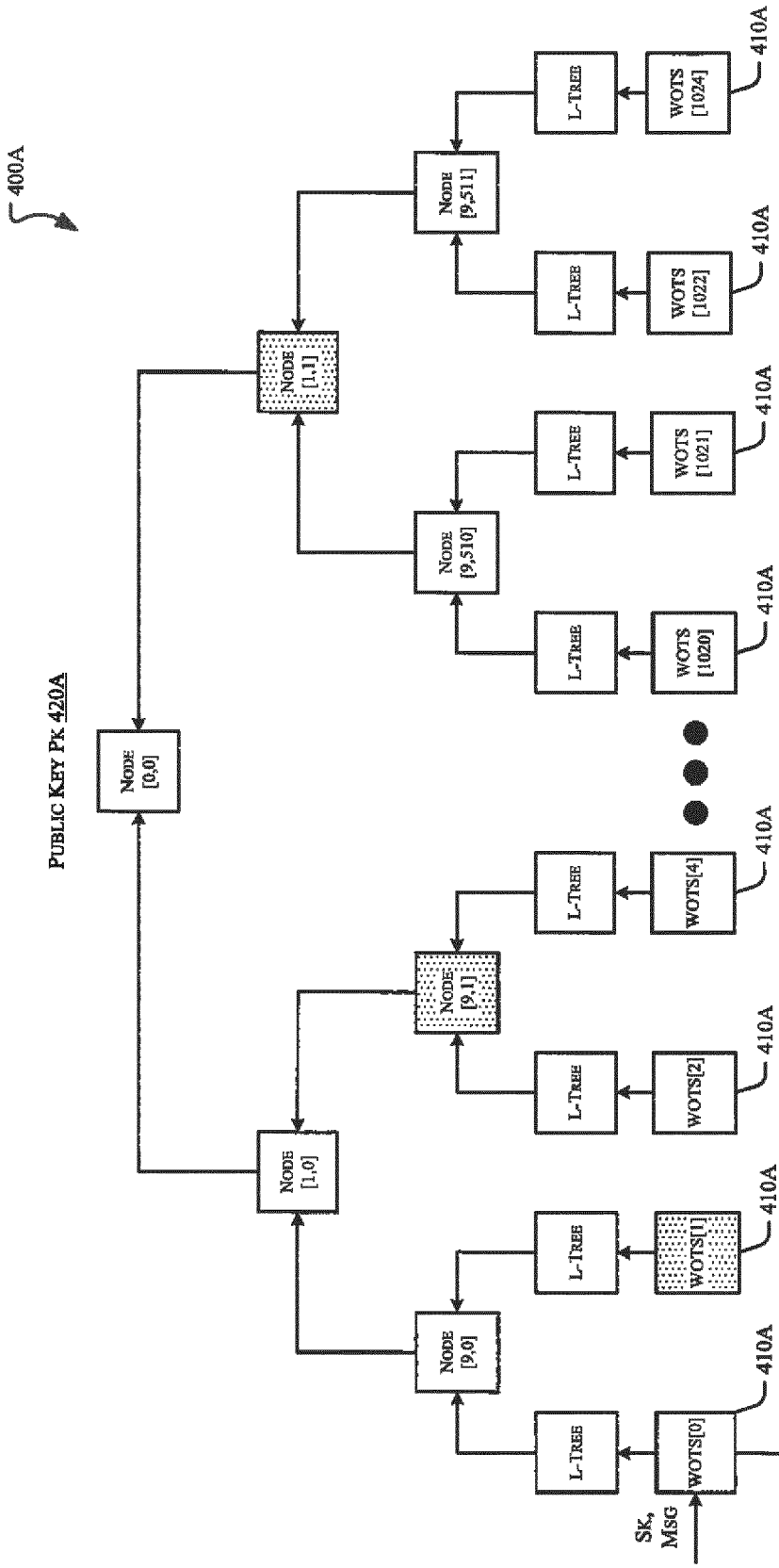
400A
PUBLIC KEY PK 420A
NODE [0,0]
NODE [1,0]
NODE [1,1]
NODE [9,0]
NODE [9,1]
NODE [9,510]
NODE [9,511]
L-TREE
WOTS[0]
WOTS[1]
WOTS[2]
WOTS[4]
WOTS [1020]
WOTS [1021]
WOTS [1022]
WOTS [1024]
410A
SK, MSG
SIGNATURE

FIG. 4A

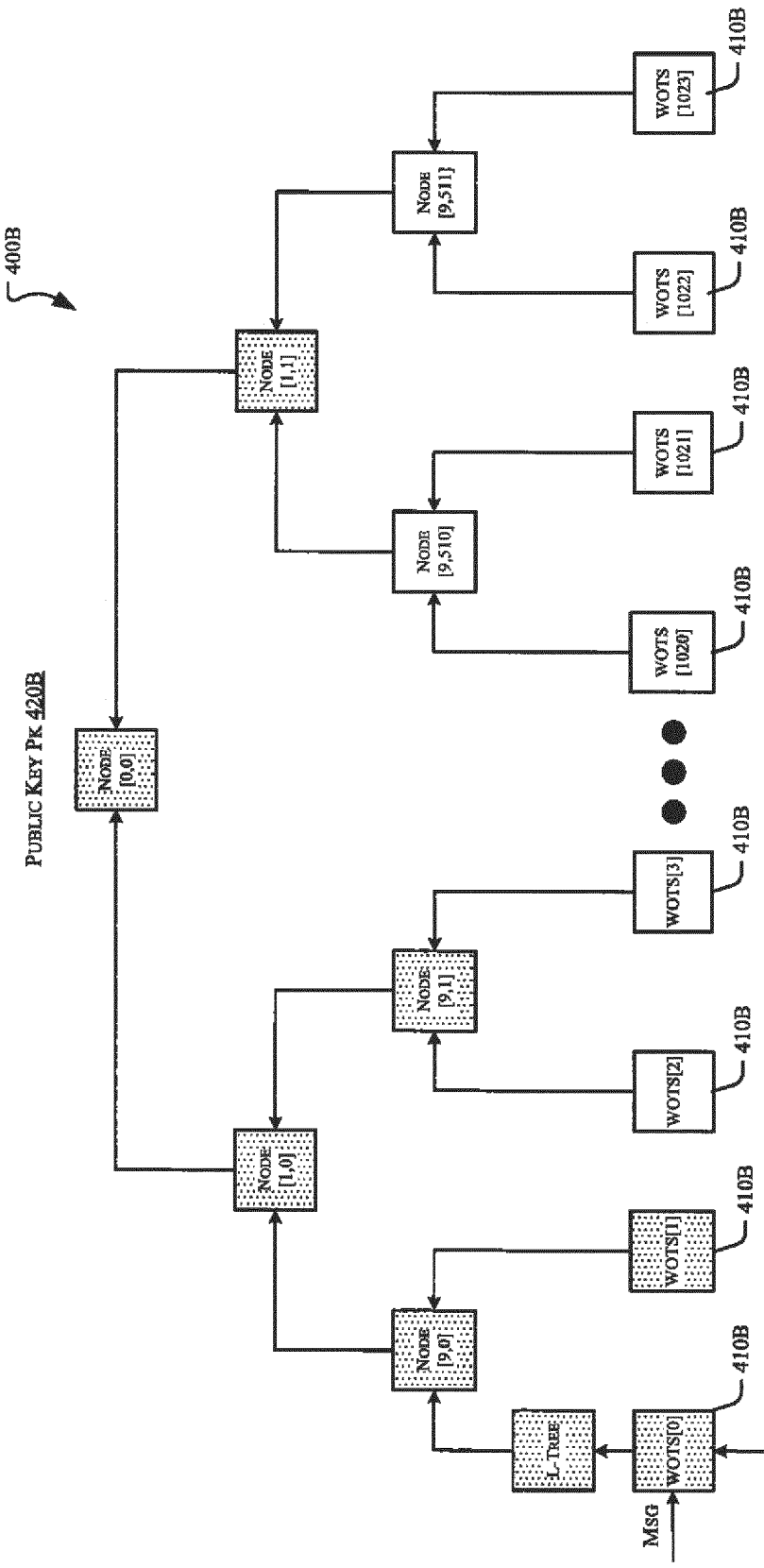
400B
PUBLIC KEY PK 420B
NODE [0,0]
NODE [1,0]
NODE [1,1]
NODE [9,0]
NODE [9,1]
NODE [9,510]
NODE [9,511]
L-TREE
WOTS[0]
WOTS[1]
WOTS[2]
WOTS[3]
WOTS [1020]
WOTS [1021]
WOTS [1022]
WOTS [1023]
410B
MSG
SIGNATURE

FIG. 4B

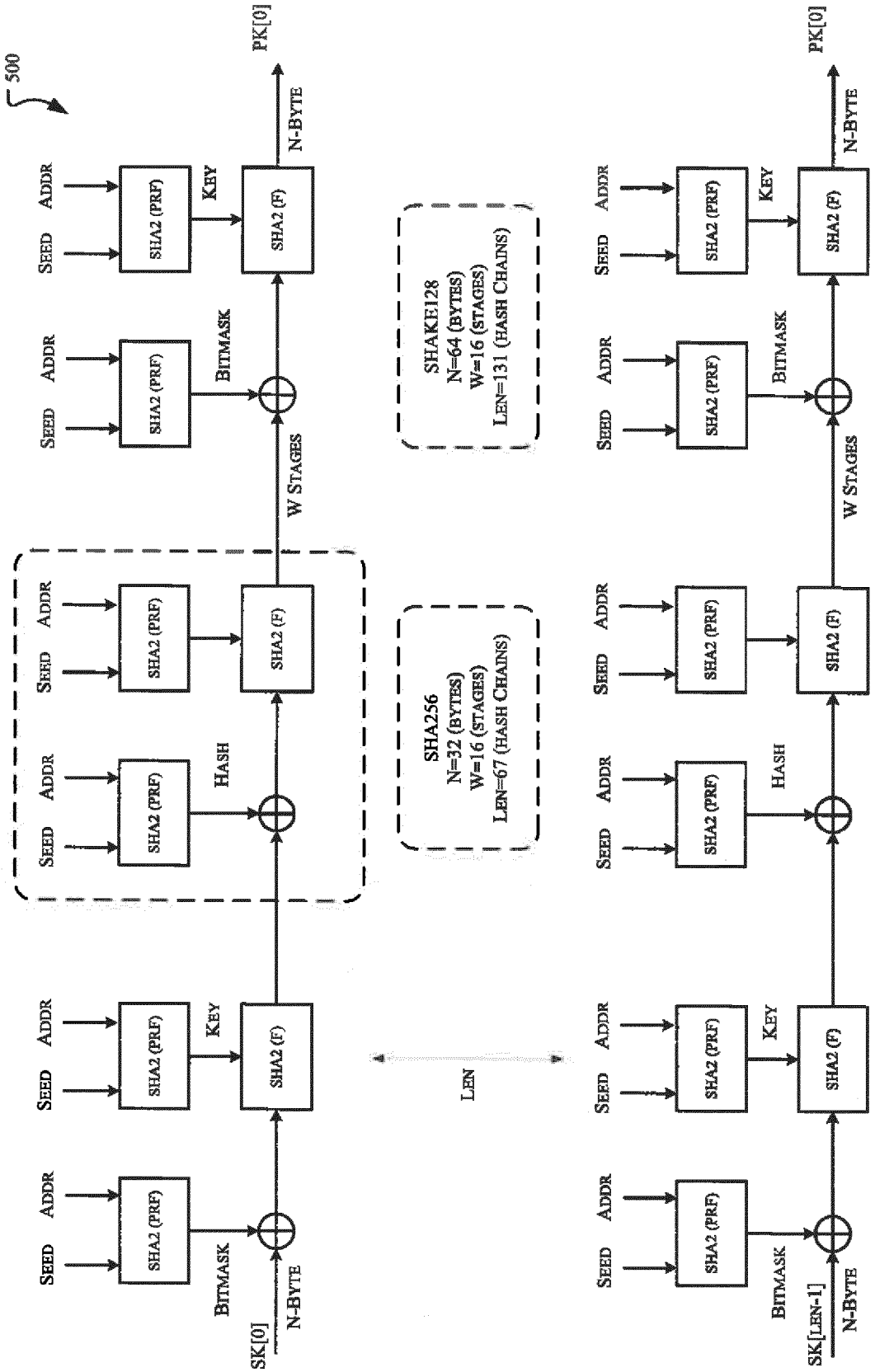
500
PK[0]
N-BYTE
KEY
SHA2 (F)
SHA2 (PRF)
SEED
ADDR
BITMASK
W STAGES
HASH
SK[0]
SK[LEN-1]
LEN
SHA256
N=32 (BYTES)
W=16 (STAGES)
LEN=67 (HASH CHAINS)
SHAKE128
N=64 (BYTES)
W=16 (STAGES)
LEN=131 (HASH CHAINS)

FIG. 5

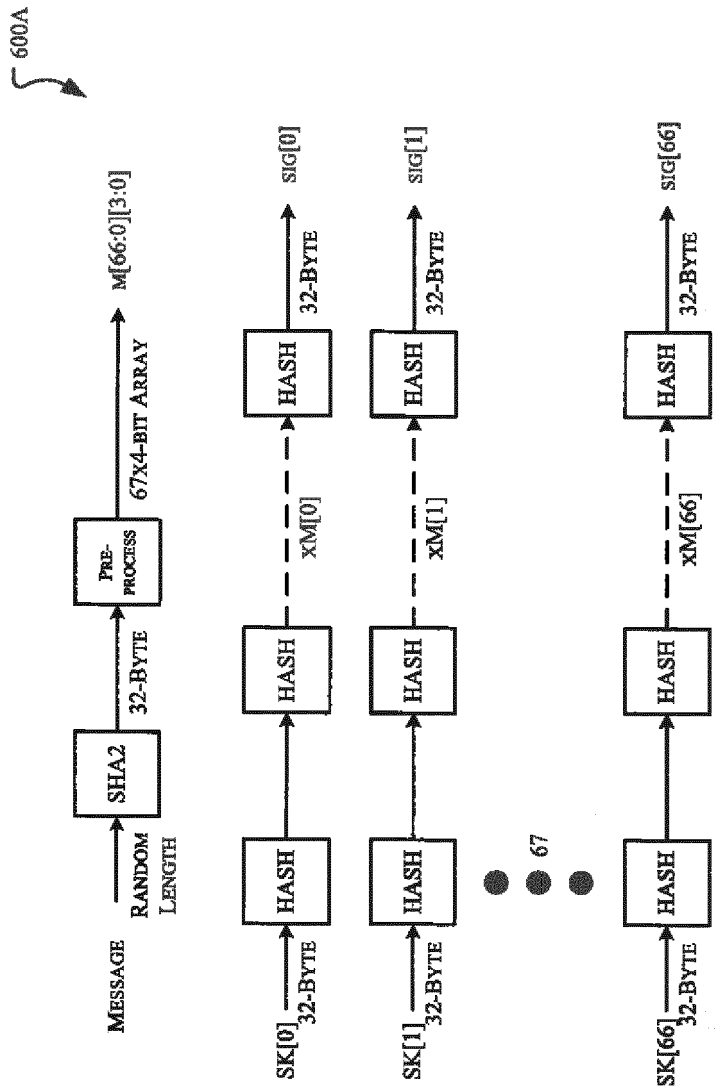
600A
MESSAGE
RANDOM LENGTH
SHA2
32-BYTE
PRE-PROCESS
67X4-BIT ARRAY
M[66:0][3:0]
SK[0]
SK[1]
SK[66]
32-BYTE
HASH
xM[0]
xM[1]
xM[66]
67
SIG[0]
SIG[1]
SIG[66]

FIG. 6A

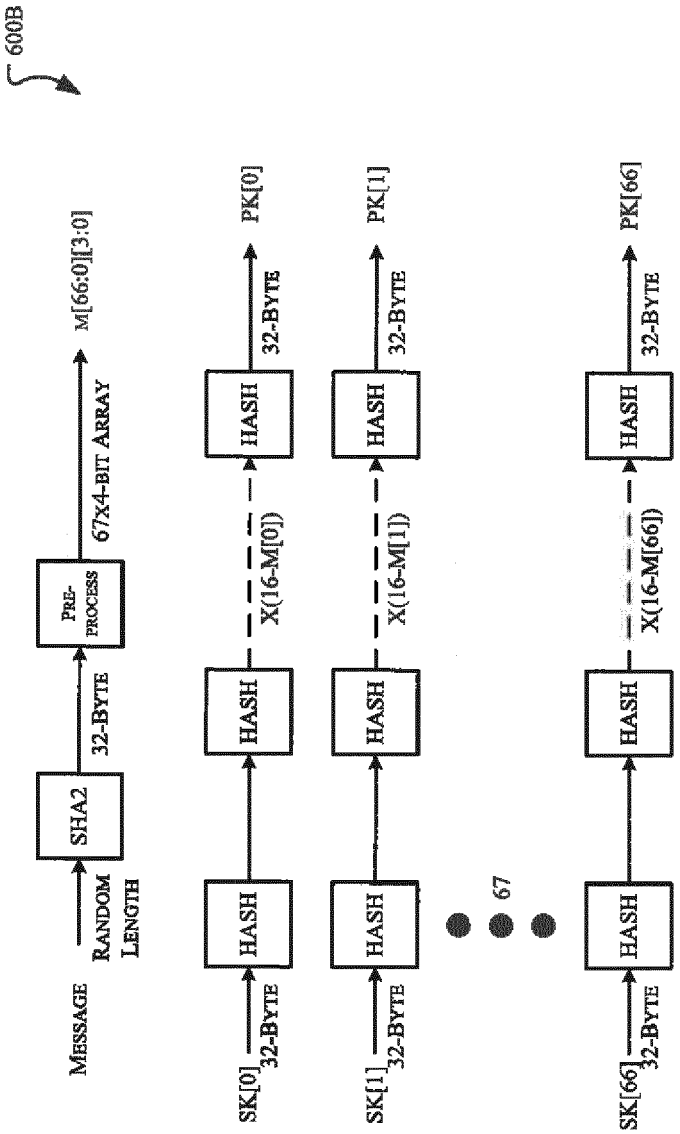
600B
MESSAGE
RANDOM
LENGTH
SHA2
32-BYTE
PRE-PROCESS
67X4-BIT ARRAY
M[66:0][3:0]
SK[0]
SK[1]
SK[66]
32-BYTE
HASH
X(16-M[0])
X(16-M[1])
X(16-M[66])
PK[0]
PK[1]
PK[66]
67

FIG. 6B

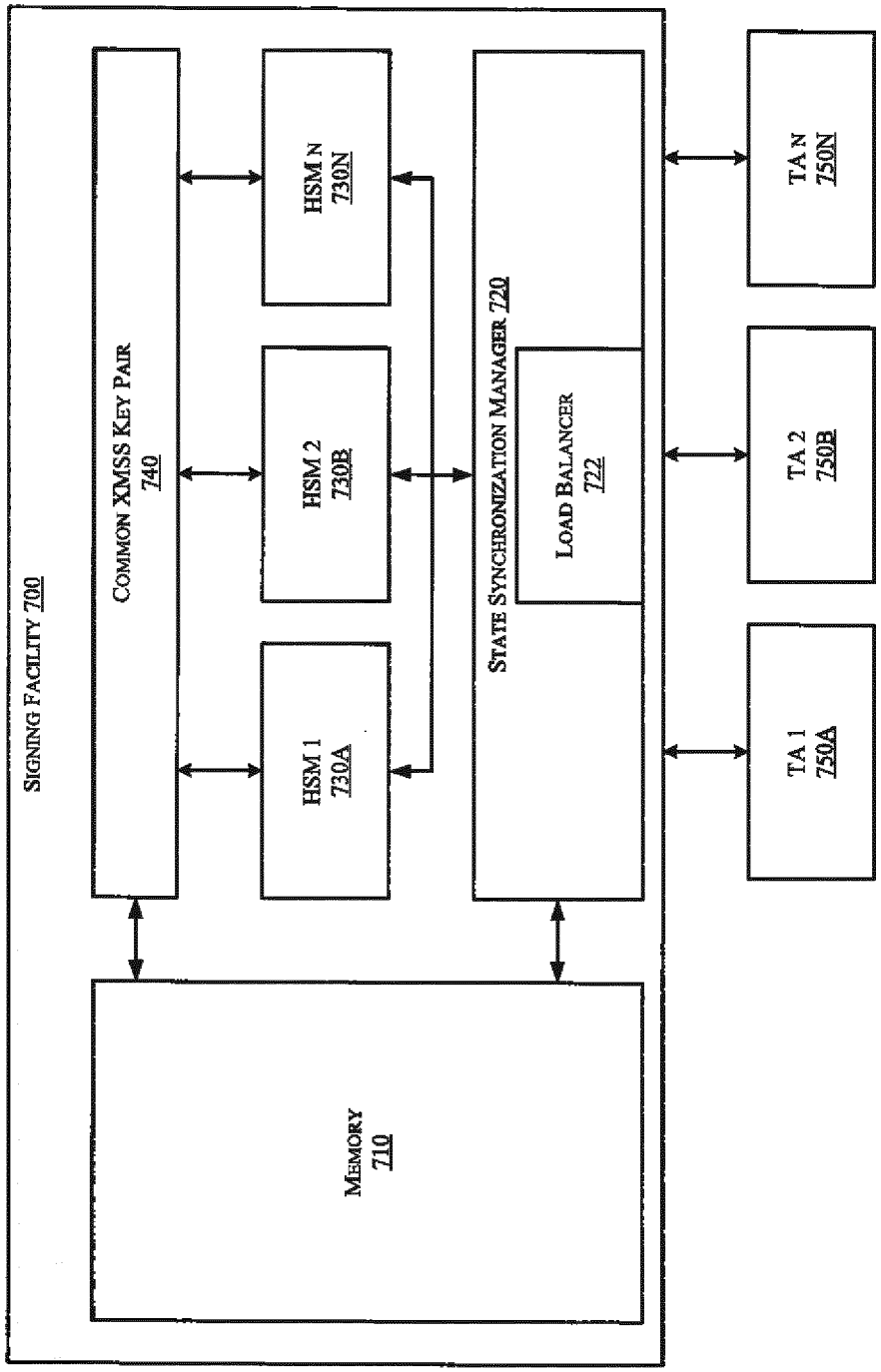
SIGNING FACILITY 700
COMMON XMSS KEY PAIR 740
HSM 1 730A
HSM 2 730B
HSM N 730N
STATE SYNCHRONIZATION MANAGER 720
LOAD BALANCER 722
MEMORY 710
TA 1 750A
TA 2 750B
TA N 750N

FIG. 7

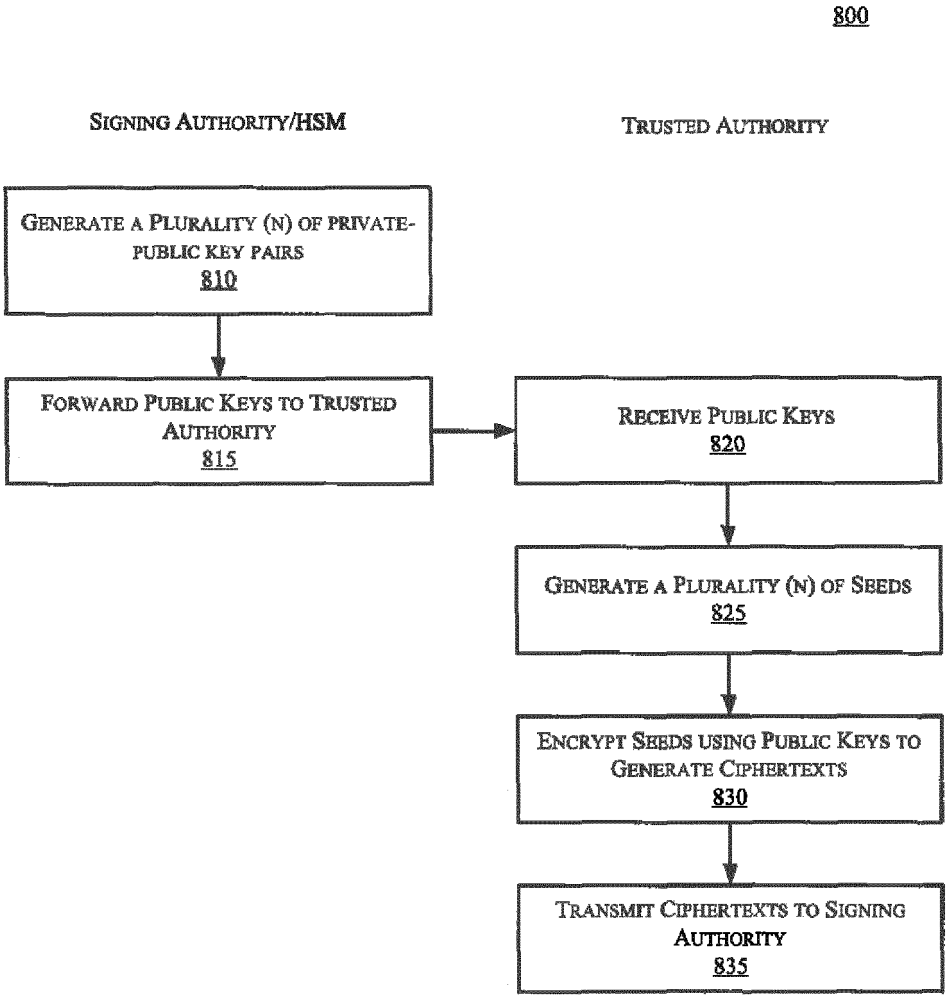
800
SIGNING AUTHORITY/HSM
TRUSTED AUTHORITY
GENERATE A PLURALITY (N) OF PRIVATE-PUBLIC KEY PAIRS
810
FORWARD PUBLIC KEYS TO TRUSTED AUTHORITY
815
RECEIVE PUBLIC KEYS
820
GENERATE A PLURALITY (N) OF SEEDS
825
ENCRYPT SEEDS USING PUBLIC KEYS TO GENERATE CIPHERTEXTS
830
TRANSMIT CIPHERTEXTS TO SIGNING AUTHORITY
835

FIG. 8

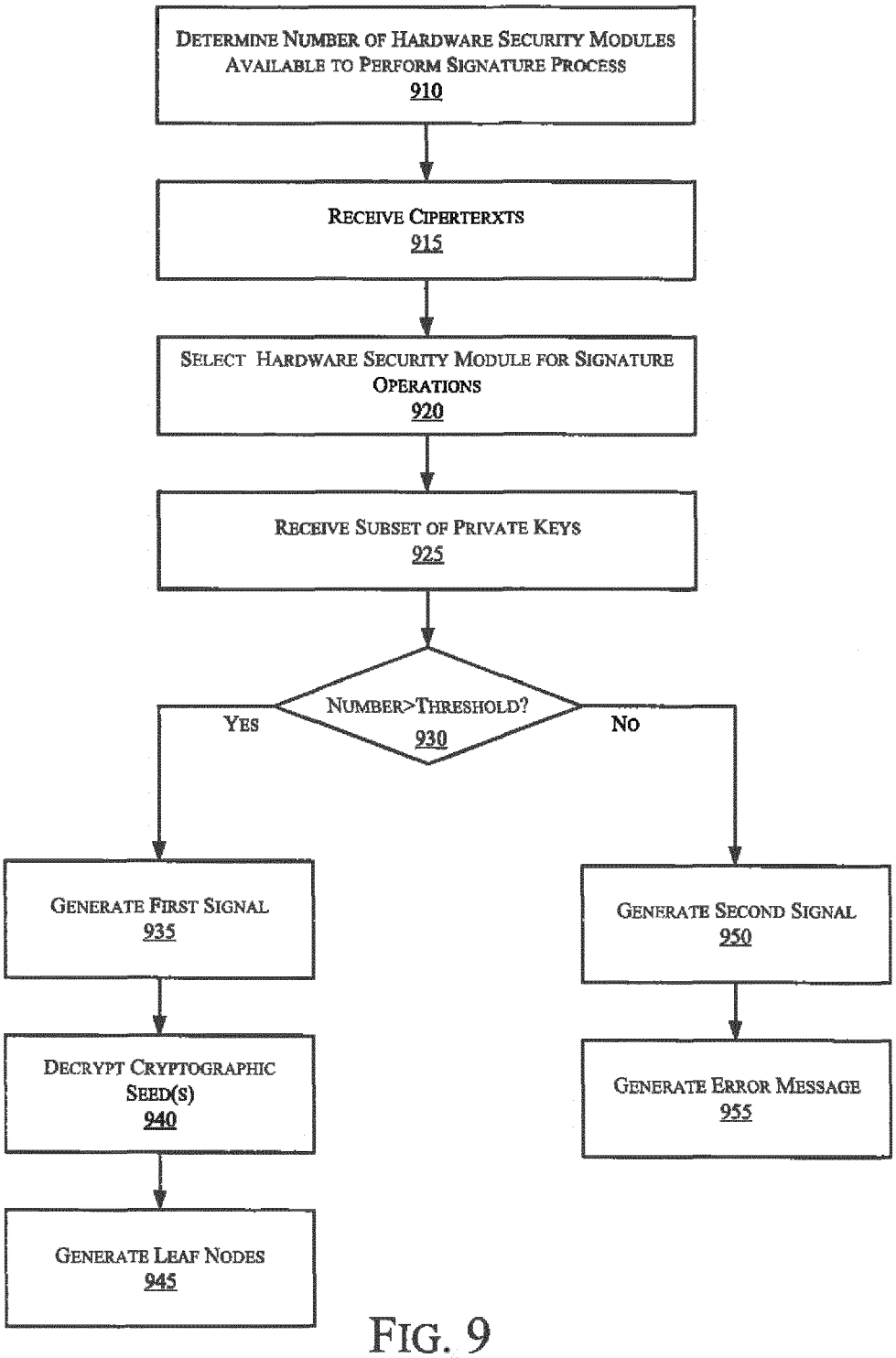
DETERMINE NUMBER OF HARDWARE SECURITY MODULES AVAILABLE TO PERFORM SIGNATURE PROCESS
910
RECEIVE CIPERTERXTS
915
SELECT HARDWARE SECURITY MODULE FOR SIGNATURE OPERATIONS
920
RECEIVE SUBSET OF PRIVATE KEYS
925
NUMBER>THRESHOLD?
930
YES
NO
GENERATE FIRST SIGNAL
935
DECRYPT CRYPTOGRAPHIC SEED(S)
940
GENERATE LEAF NODES
945
GENERATE SECOND SIGNAL
950
GENERATE ERROR MESSAGE
955

FIG. 9

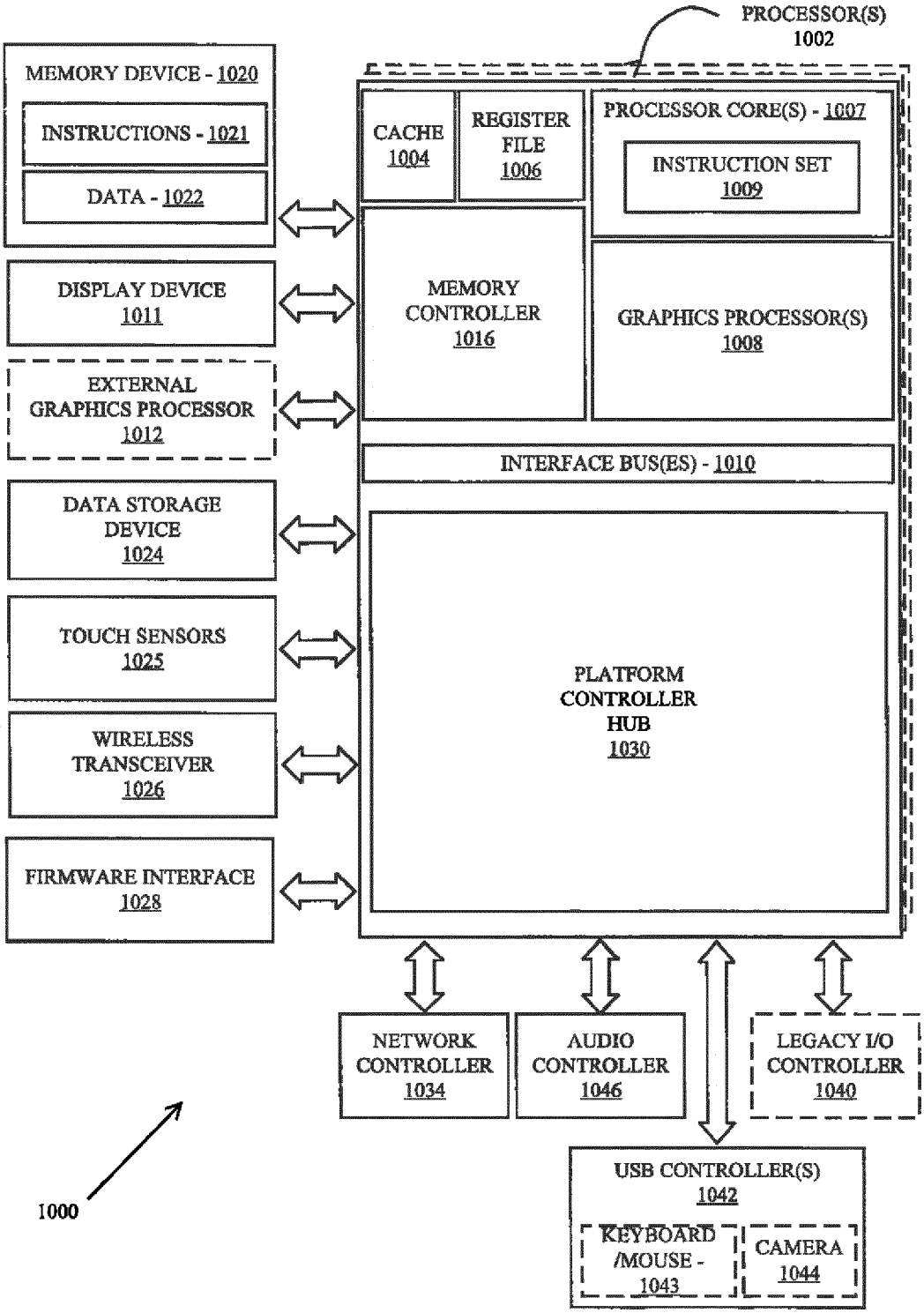
PROCESSOR(S)
1002
MEMORY DEVICE - 1020
INSTRUCTIONS - 1021
DATA - 1022
CACHE
1004
REGISTER
FILE
1006
PROCESSOR CORE(S) - 1007
INSTRUCTION SET
1009
DISPLAY DEVICE
1011
MEMORY
CONTROLLER
1016
GRAPHICS PROCESSOR(S)
1008
EXTERNAL
GRAPHICS PROCESSOR
1012
INTERFACE BUS(ES) - 1010
DATA STORAGE
DEVICE
1024
TOUCH SENSORS
1025
PLATFORM
CONTROLLER
HUB
1030
WIRELESS
TRANSCEIVER
1026
FIRMWARE INTERFACE
1028
NETWORK
CONTROLLER
1034
AUDIO
CONTROLLER
1046
LEGACY I/O
CONTROLLER
1040
USB CONTROLLER(S)
1042
KEYBOARD
/MOUSE -
1043
CAMERA
1044
1000

FIG. 10

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 21 18 6074

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	EP 3 672 143 A1 (SAFENET CANADA INC [CA]; THALES DIS FRANCE SA [FR]) 24 June 2020 (2020-06-24) * abstract * * paragraph [0003] - paragraph [0027] * * paragraph [0041] - paragraph [0091] *	1-14	INV. H04L9/32 H04L9/08
A	US 2020/220735 A1 (PRAUS SLAVKA [US] ET AL) 9 July 2020 (2020-07-09) * abstract * * paragraph [0034] - paragraph [0035] * * paragraph [0066] - paragraph [0100] *	1-14	
A	WO 2019/202314 A1 (R3 LTD [GB]) 24 October 2019 (2019-10-24) * abstract * * paragraph [0011] - paragraph [0012] * * paragraph [0038] - paragraph [0061] *	1-14	
A	US 2018/367316 A1 (CHENG GANG [US] ET AL) 20 December 2018 (2018-12-20) * paragraph [0177] * * paragraph [0227] *	1-14	TECHNICAL FIELDS SEARCHED (IPC) H04L

1 The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	9 December 2021	Apostolescu, Radu

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 21 18 6074

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

09-12-2021

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 3672143 A1	24-06-2020	EP 3672143 A1	24-06-2020
		EP 3900256 A1	27-10-2021
		WO 2020126235 A1	25-06-2020
US 2020220735 A1	09-07-2020	US 10608824 B1	31-03-2020
		US 2020220735 A1	09-07-2020
WO 2019202314 A1	24-10-2019	EP 3782326 A1	24-02-2021
		US 2019319798 A1	17-10-2019
		US 2021176072 A1	10-06-2021
		WO 2019202314 A1	24-10-2019
US 2018367316 A1	20-12-2018	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Non-patent literature cited in the description

- Secure Hash Standard (SHS). Federal Information Processing Standards (FIPS). National Institute of Standards and Technology (NIST), March 2012 **[0025] [0034]**
- SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. FIPS. NIST, August 2015 **[0025] [0034]**
- XMSS: Extended Hash-Based Signatures. Internet Research Task Force, April 2015 **[0027]**